

Purdue University

Purdue e-Pubs

Department of Computer Science Technical
Reports

Department of Computer Science

1993

Rowdlas User's Guide

H. Byun

Elias N. Houstis

Purdue University, enh@cs.purdue.edu

E. A. Vavalis

Report Number:

93-017

Byun, H.; Houstis, Elias N.; and Vavalis, E. A., "Rowdlas User's Guide" (1993). *Department of Computer Science Technical Reports*. Paper 1035.
<https://docs.lib.purdue.edu/cstech/1035>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

ROWDLAS USER'S GUIDE

**H. Byun
E. N Houstis
E. A. Vavalis**

**CSD-TR-93-017
February 1993**

ROWDLAS USER'S GUIDE

H. BYUN AND E.N. HOUSTIS AND E.A. VAVALIS
PURDUE UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
WEST LAFAYETTE, IN 47907

Abstract

This report is a user's guide to ROWDLAS, a Row-oriented Distributed Linear Algebra Subroutine package. ROWDLAS can be used to develop higher level linear algebra blocks (like direct and iterative solvers) on distributed multiprocessor systems. It also provides portability across several distributed-memory multiprocessors and execution tracing for monitoring the performance and for debugging purposes.

1. Introduction. ROWDLAS is a collection of FORTRAN routines that perform basic linear algebra operations on distributed memory systems. They assume that matrices are row distributed among processors. Only information concerning the data distribution, and the interconnection topology, is required while all other architectural details are hide from the user.

For both efficiency and portability ROWDLAS is built on top of BLAS [7], [3], [2] and PCL [4] BLAS routines are used at the processor level to perform local linear algebra operations and PCL high-level communication routines are used to accumulate global information. Build-in execution tracing for monitoring the performance is available through PCL and PARAGRAPH [5]. Some preliminary data on the performance evaluation of ROWDLAS routines, together with the detailed description of the parallel algorithms used, can be found in [1].

Some of the ROWDLAS routines assume certain distribution, mapping and storage schemes for the data they involve. This information can be found in section 2. In section 3 we give detailed descriptions of the ROWDLAS routines and their arguments.

2. Assumptions and requirements. Parallel distributed memory algorithms and routines are based on certain architectural parameters like the number of processors, the interconnection topology, the numbering of the node processors and the mapping of the data onto the architecture. In ROWDLAS we have tried to make the definition of these parameters as easy as possible. In this section we list the available choices for these parameters and describe the mechanisms to assign values to them.

Before the call of any of the ROWDLAS routines the users should:

- Call the PCL routine *open0* to specify the number of processors *nprocs*, the host and node ids, *me* and *host* respectively and enable communication. These parameters should be stored in the common block
`common /open/ nprocs, me, host`
- Call the PCL routine *setarc0* to specify the interconnection topology and the numbering of the nodes. *setarc0* can be called more than once in case one needs to modify the topology or the node numbering according to the needs of the ROWDLAS routines.
- Some routines require information on the distribution of the data. In this case the user should provide the array *idst*. *idst(i)*, $i = 1, \dots, nprocs + 1$ hold the global index of the first element of the data that belongs to processor *i*.

In the following table we list the possible choices for the above mentioned parameters for each ROWDLAS routine.

The user simply links the ROWDLAS, and PICL libraries together with his favored uni-processor BLAS routines, like the highly optimized ones in [6].

3. Routine Descriptions. In this section we describe the Linear Algebra Sub-programs currently available in ROWDLAS. We use the BLAS naming convention and the argument lists are essential the BLAS ones with the only exception of the *idst* array that hold the data distribution information.

The routines are arranged in alphabetical order and all of them come in single and double precision real.

dsamin/ddamin

Syntax: $w = \text{dsamin/ddamin}(n, x, \text{idst})$

Purpose: Finds the smallest (in absolute value) element of the vector x .

$$w = \min_{1 \leq i \leq n} (|x_i|)$$

On Entry

n	integer the order of the vector x .
x	Single precision real for dsamin Double precision real for ddamin array of dimension n
idst	integer array of dimension $nprocs$ array holding the distribution information of x .

On Return

w	single precision real for dsamin single precision real for ddamin the absolute value of the smallest (in absolute value) element of x .
-----	-----------------------------------------------------------------------------------------------------------------------------------------------------

dsamax/ddamax

Syntax: $w = \text{dsamax/ddamax}(n, x, \text{idst})$

Purpose: Finds the largest (in absolute value) element of the vector x .

$$w = \max_{1 \leq i \leq n} (|x_i|)$$

On Entry

n	integer the order of the vector x .
x	Single precision real for dsamax Double precision real for ddamax array of dimension n
idst	integer array of dimension n_{procs} array holding the distribution information of x .

On Return

w	single precision real for dsamax double precision real for ddamax the absolute value of the largest (in absolute value) element of x .
-----	----------------------------------------------------------------------------------------------------------------------------------------------------

idsamin/iddamin

Syntax: `iw = idsamin/iddamin(n,x,idst)`

Purpose: Finds the smallest index i such that:

$$|x_i| = \min_{1 \leq i \leq n} (|x_j|), j = 1, \dots, n$$

On Entry

n	integer the order of the vector x .
x	Single precision real for idsamin Double precision real for iddamin array of dimension n
idst	integer array of dimension $nprocs$ array holding the distribution information of x .

On Return

iw	integer index of smallest (in absolute value) element of x .
----	-----------------------------------------------------------------------

idsamax/iddamax

Syntax: `iw = idsamax/iddamax(n,x,idst)`

Purpose: Finds the largest index i such that:

$$|x_i| = \max_{1 \leq j \leq n} (|x_j|), j = 1, \dots, n$$

On Entry

n	integer the order of the vector x .
x	Single precision real for <code>idsamax</code> Double precision real for <code>iddamax</code> array of dimension n
idst	integer array of dimension $nprocs$ array holding the distribution information of x .

On Return

iw	integer index of largest (in absolute value) element of x .
----	----------------------------------------------------------------------

dsasum/ddasum

Syntax: $w = \text{dsasum/ddasum}(n, x, \text{idst})$

Purpose: Computes the sum of magnitudes of the elements of the x :

$$w = \sum_{i=1}^n |x_i|$$

On Entry

n	integer the order of the vector x .
x	Single precision real for dsasum Double precision real for ddasum array of dimension n
idst	integer array of dimension n_{procs} array holding the distribution information of x .

On Return

w	single precision real for dsasum double precision real for ddasum sum of magnitudes of the elements of x .
-----	------------------------------------------------------------------------------------------------------------------------

dsnrm2/ddnrm2

Syntax: $w = \text{dsnrm2/ddnrm2}(n, x, \text{idst})$

Purpose: Computes the *Euclidean norm* of the vector x :

$$w = \left(\sum_{i=1}^n x_i^2 \right)^{1/2}$$

On Entry

n	integer the order of the vector x .
x	Single precision real for dsrm2 Double precision real for ddrm2 array of dimension n
idst	integer array of dimension $nprocs$ array holding the distribution information of x .

On Return

w	single precision real for dsrm2 double precision real for ddrm2 Euclidean norm of x .
---	---------------------------------------------------------------------------------------------------

dsdot/dddots/ddsdot

Syntax: $w = \text{dsdot/dddots/ddsdot}(n,x,y,\text{idst})$

Purpose: Computes the dot product of two vectors:

$$w = x^T y$$

dddots takes single precision arguments, but performs the summation in double precision and returns a double precision result.

On Entry

n	integer the order of the vector x .
x	<i>LOCAL</i> single precision real for dsdot and ddsdot <i>LOCAL</i> double precision real for ddddots array of dimension n
y	Single precision real for dsdot and ddsdot Double precision real for ddddots array of dimension n
idst	integer array of dimension n_{procs} array holding the distribution information of y .

On Return

w	single precision real for dsdot double precision real for ddddots and ddsdot the dot product of x and y .
-----	-------------------------------------------------------------------------------------------------------------------------

dsgemv/ddgemv

Syntax: $w = \text{dsgemv/ddgemv}(\text{trans}, m, n, \alpha, a, lda, x, \beta, y, idst)$

Purpose: Performs one of the matrix-vector operations

$$y = \alpha Ax + \beta y$$

$$y = \alpha A^T x + \beta y$$

where α and β are scalars, x and y are vectors and A is an m by n matrix.

On Entry

trans	character*1
	specifies the operation to be performed as follows trans = 'N' or 'n' $y = \alpha Ax + \beta y$ trans = 'T' or 't' $y = \alpha A^T x + \beta y$
m	integer
	the number of rows of the matrix a .
n	integer
	the number of columns of the matrix a .
alpha	Single precision for dsgemv Double precision for ddgemv
	specifies the scalar α
a	Single precision for dsgemv Double precision for ddgemv
	array of dimension (lda, n) . Contains the matrix on its leading m by n part
lda	integer
	the first dimension of a as declared in the calling (sub)program. must be at least $\max(1, m)$
x	Single precision for dsgemv Double precision for ddgemv
	array of dimension n if $trans = 'n'$ or m if $trans = 't'$.

beta	Single precision for dsgev Double precision for ddgev specifies the scalar β
y	Single precision for dsgev Double precision for ddgev array of dimension n if $trans = 'n'$ or m if $trans = 't'$.
idst	integer array of dimension $nprocs$ array holding the distribution information of A , x and y .
On Return	
y	overwritten by the updated vector y.

dsbmv/ddbmv

Syntax: $w = \text{dsbmv/ddbmv}(\text{trans}, m, n, kl, ku, \alpha, a, lda, x, \beta, y, idst)$

Purpose: Performs one of the matrix-vector operations

$$y = \alpha Ax + \beta y$$

$$y = \alpha A^T x + \beta y$$

where α and β are scalars, x and y are vectors and A is an m by n banded matrix with lower bandwidth kl and upper bandwidth ku .

On Entry

trans	character*1
	specifies the operation to be performed as follows trans = 'N' or 'n' $y = \alpha Ax + \beta y$ trans = 'T' or 't' $y = \alpha A^T x + \beta y$
m	integer
	the number of rows of the matrix a .
n	integer
	the number of columns of the matrix a .
kl	integer
	the number of sub-diagonals of the matrix a .
ku	integer
	the number of super-diagonals of the matrix a .
alpha	Single precision for dsbmv Double precision for ddbmv
	specifies the scalar α
a	Single precision for dsbmv Double precision for ddbmv
	array of dimension (lda, n) . Contains the matrix on its leading m by n part The matrix is locally stored in standard (not banded) form

lda	integer
	the first dimension of a as declared in the calling (sub)program. must be at least $\max(1, m)$
x	Single precision for dsbmv Double precision for ddbmv
	array of dimension n if $trans = 'n'$ or m if $trans = 't'$.
beta	Single precision for dsbmv Double precision for ddbmv
	specifies the scalar β
y	Single precision for dsbmv Double precision for ddbmv
	array of dimension n if $trans = 'n'$ or m if $trans = 't'$.
idst	integer array of dimension $nprocs$
	array holding the distribution information of A , x and y .
On Return	
y	overwritten by the updated vector y.

NOTES

- On each processor, only the non-zero *BLOCKS* of the matrix A are stored.
- The distributed data should be mapped assuming ring or full connectivity and the *setarc0* should be called accordingly.

dssemv/ddsemv

Syntax: $w = \text{dssemv/ddsemv}(\text{trans}, m, n, \alpha, \text{coef}, \text{idcoef}, \text{lda}, x, \beta, y, \text{idst})$

Purpose: Performs one of the matrix-vector operations

$$y = \alpha Ax + \beta y$$

$$y = \alpha A^T x + \beta y$$

where α and β are scalars, x and y are vectors and A is an m by n sparse matrix stored in a compressed form using the arrays *coef* and *idcoef*.

On Entry

trans	character*1
	specifies the operation to be performed as follows trans = 'N' or 'n' $y = \alpha Ax + \beta y$ trans = 'T' or 't' $y = \alpha A^T x + \beta y$
m	integer
	the number of rows of the matrix a .
n	integer
	the number of columns of the matrix a .
alpha	Single precision for dssemv Double precision for ddsemv
	specifies the scalar α
coef	Single precision for dssemv Double precision for ddsemv
	array of dimension (lda, n) . Contains the coefficients of the matrix A on its leading m by n part
idcoef	Single precision for dssemv Double precision for ddsemv
	array of dimension (lda, n) . Contains the indices of the coefficients of the matrix A on its leading m by n part

lda	integer
	the first dimension of <i>coef</i> and <i>idcoef</i> as declared in the calling (sub)program, must be at least $\max(1, m)$
x	Single precision for dssemv Double precision for ddsemv
	array of dimension n if <i>trans</i> = 'n' or m if <i>trans</i> = 't'.
beta	Single precision for dssemv Double precision for ddsemv
	specifies the scalar β
y	Single precision for dssemv Double precision for ddsemv
	array of dimension n if <i>trans</i> = 'n' or m if <i>trans</i> = 't'.
idst	integer array of dimension <i>nprocs</i>
	array holding the distribution information of A, x and y .

On Return

y	overwritten by the updated vector y.
---	--------------------------------------

NOTES

- The distributed data should be mapped assuming ring or full connectivity and the *setarc0* should be called accordingly.

REFERENCES

- [1] H. BYUN, E. HOUSTIS, AND E. VAVALIS, *Some experiments with a set of linear algebra routines for distributed memory parallel systems*, Tech. Report CSTR-222, Purdue University, W. Lafayette, IN, 1992.
- [2] J. J. DONGARRA, J. DU CROZ, I. DUFF, AND S. HAMMARLING, *A set of level 3 basic linear algebra subprograms*.
- [3] J. J. DONGARRA, J. DU CROZ, S. HAMMARLING, AND R. J. HANSON, *An extended set of basic linear algebra subprograms: Model implementation and test programs*, ACM Trans. Math. Softw., 14 (1988), pp. 18-32.
- [4] G. GEIST, M. HEATH, B. PEYTON, AND P. WORLEY, *A users' guide to PICL a portable instrumented communication library*, Tech. Report ORNL/TM-11616, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN, Oct. 1990.
- [5] M. HEATH AND J. ETHERIDGE, *Visualizing the performance of parallel programs*, IEEE Software, 8 (1991), pp. 29-39.
- [6] INTEL CORPORATION, *iPSC/860 Basic Math Library User's Guide*, 1 ed., April 1991.
- [7] C. L. LAWSON, R. J. HANSON, D. R. KINCAID, AND F. T. KROGH, *Basic linear algebraic subprogram for Fortran usage*, ACM Trans. Math. Softw., 5 (1979), pp. 324-325.